ResNet-Based Image Classification for CIFAR-10

Yin Wufei
New York University
Github: https://github.com/ZBW-P/Deep-learning-resnet.git

Abstract

The residual network is easy to optimize and can improve accuracy by increasing the depth. The residual block inside uses skip connections to alleviate the gradient vanishing problem caused by increasing the depth in deep neural networks. In this project, we designed and optimized an improved residual network architecture to achieve high accuracy on image classification datasets while keeping the constraint of no more than 5 million parameters. The main directions of our optimization efforts are: regularization, model structure optimization, data enhancement, and learning rate adjustment.

Introduction

In this project, we implement a deep learning based image classification model using ResNet and Squeeze and Excitation (SE) blocks to enhance feature learning. The model is trained and evaluated on the CIFAR-10 dataset, which contains 60,000 RGB images of size 32×32 from 10 categories. Our approach integrates advanced data augmentation techniques, regularization strategies, and adaptive learning rate schedules to improve generalization and robustness.

Squeeze and Excitation (SE) Block:

We incorporate SE blocks into ResNet to enhance channel feature representation. SE blocks utilize global average pooling to capture channel dependencies and apply dynamic recalibration, enabling the model to focus on more informative channels.

Data Augmentation:

To increase data diversity and improve model robustness, we apply several augmentation techniques including:

- Random Cropping: Increase spatial variations by randomly cropping the input image.
- Random Horizontal Flip: Increase invariance to left-right transformations.
- Color Jittering: Adjusts brightness, contrast, saturation, and hue to enhance generalization.
- Cutout: Randomly masks a portion of the image to encourage the model to learn distributed feature representations.

Mixup Augmentation:

- We implement mixup augmentation, which inserts pairs of training samples and their labels to create new synthetic examples. This reduces overfitting and improves robustness to adversarial.

Regularization Strategies:

- Dropout: Applied before fully connected layers, prevents overfitting by randomly deactivating neurons during training.

Adaptive Learning Rate Scheduling:

- Warm-up Strategy: The learning rate starts with a small value and is gradually increased over the first 15 epochs to stabilize the training.
- Cosine Annealing: After the warm-up phase, the learning rate is cosine annealed for the remaining epochs, resulting in smooth convergence while avoiding sharp decay.

Training Strategy:

- The model is trained for 250 epochs with a batch size of 128 and optimized using SGD with momentum.
- Label smoothing is applied to mitigate overconfidence in predictions.

The final prediction results show that the model significantly improves the classification performance on CIFAR-10. The integration of these technologies enables the model to achieve better generalization, robustness and convergence stability. The final prediction accuracy for the unlabeled dataset reached 0.85574.

This project demonstrates the most advanced deep learning techniques in image classification and emphasizes the importance of data augmentation, feature recalibration, regularization and adaptive optimization in training residual networks.

Technology Used

We employed a modified ResNet architecture combined with Squeeze-and-Excitation (SE) mechanism to achieve efficient image classification under parameter constraints. Below is a detailed description of our technical implementation:

Model Architecture Design

Lightweight SE-ResNet Architecture Our core network architecture is based on ResNet, but with several optimizations to reduce parameter count while maintaining performance:

1. **Channel Configuration Optimization**:
   - Adopted a progressive channel design of {32-64-128-256}, significantly lower than standard ResNet-18's {64-128-256-512}.
   - This configuration maintains hierarchical feature extraction capability while substantially reducing model parameters.
2. **Residual Block Design**:
   - Each residual block contains two 3×3 convolutional layers, each followed by batch normalization and ReLU activation.
   - Classic skip connection design ensures smooth gradient flow and faster convergence.
   - 1×1 convolutions used for dimension adjustment when needed.
3. **SE Attention Mechanism**:
   - SE modules integrated in deeper layers (layer3 and layer4), enabling the network to dynamically adjust channel weights.
   - SE modules use global average pooling to obtain channel descriptors, followed by non-linear transformations to produce importance weights for each channel.
   - 16:1 reduction ratio used for parameter efficiency, applying SE modules only in critical layers to balance performance and complexity.
4. **Overall Network Structure**: ResNet (Residual Network) is a deep learning architecture designed to improve gradient flow and prevent vanishing gradients through residual connections. These connections allow the network to learn residual mappings instead of directly fitting transformations. The mathematical representation of a residual block is:

$$Y = \text{ReLU}(X + F(X)) \tag{1}$$

where X is the input, and $F(X)$ represents the transformation applied within the block.

Implementation of ResNet-18 Structure

Our group first defined the basic building block of ResNet-18, which consists of:

- Two convolutional layers.
- Two batch normalization (BN) layers.
- One ReLU activation.
- A residual connection that adds the original input to the transformed output before the final activation.

The residual block follows the structure:

$$\text{Conv} \rightarrow \text{BN} \rightarrow \text{ReLU} \rightarrow \text{Conv} \rightarrow \text{BN} \rightarrow +F(X) \rightarrow \text{ReLU} \tag{2}$$

During implementation, we encountered a challenge where the input and output dimensions differed due to convolution operations. The height and width transformation follows the formula:

$$H_{\text{out}} = \frac{H_{\text{in}} + 2P - K}{S} + 1 \tag{3}$$

where:

- H_{in} is the input height.
- K is the kernel size.
- S is the stride.
- P is the padding.

To ensure the dimensions match when channel sizes change, we applied a 1×1 convolution with the same stride as the convolution layers.

Final Network Architecture

Our group optimized the ResNet-18 architecture by adjusting the number of layers and output channels to balance accuracy and prevent overfitting. The final structure is as follows:

- **Input Layer**: 3×3 convolution ($3 \rightarrow 32$ channels) with batch normalization and ReLU.
- **Layer 1**: 2 standard residual blocks ($32 \rightarrow 32$ channels).
- **Layer 2**: 2 standard residual blocks ($32 \rightarrow 64$ channels) with stride 2.
- **Layer 3**: 2 SE residual blocks ($64 \rightarrow 128$ channels) with stride 2.
- **Layer 4**: 2 SE residual blocks ($128 \rightarrow 256$ channels) with stride 2.
- **Global Average Pooling**.
- **Dropout** (p=0.2) to reduce overfitting.
- **Fully Connected Layer** ($256 \rightarrow 10$) for classification.

Figure 1 below shows the full route of generating the outputs from inputs.

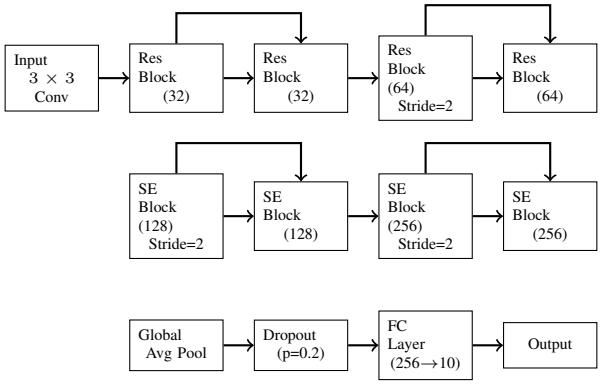

Figure 1: Optimized ResNet Architecture

Training Optimization Strategy

Data Augmentation Techniques To improve model generalization and robustness, we implemented multiple data augmentation strategies:

1. **Basic Image Augmentation**:
   - Random Cropping: 32×32 images with 4-pixel padding.
   - Random Horizontal Flipping: increases invariance to left-right flips.
   - Color Jittering: brightness=0.2, contrast=0.2, saturation=0.2, hue=0.05.
2. **Advanced Augmentation Techniques**:
   - **Cutout**: Randomly masks a 16×16 region of images, forcing the model to learn more robust feature representations.
   - **Mixup**: Linearly combines two images and their corresponding labels, smoothing decision boundaries.
   - Adaptive Mixup intensity: Gradually decreases Mixup strength (α from 0.4 to 0.1) as training progresses, balancing early regularization with later precision.

Optimizer Configuration We used the following optimizer strategies to effectively train the model:

1. **SGD Optimizer**:
   - Momentum: 0.9.
   - Weight decay: 5e-4, serving as additional regularization against overfitting.
2. **Learning Rate Strategy**:
   - "Warmup + Cosine Annealing" learning rate scheduling.
   - Warmup phase: First 15 epochs, learning rate linearly increases from near 0 to 0.1.
   - Cosine Annealing phase: Remaining 235 epochs, learning rate smoothly decays from 0.1 to 0.001.
   - Total training cycles: 250 epochs.
3. **Loss Function**:
   - Cross-entropy loss with label smoothing (smoothing=0.08).
   - Combined with Mixup for further regularization through mixed loss.
4. **Regularization Techniques**:
   - Dropout: p=0.2, applied before the fully connected layer.
   - Weight decay: Prevents excessive weight values, controlling model complexity.
   - Label smoothing + Mixup: Combined use of multiple regularization strategies significantly improves model generalization.

Our model achieves a good balance between parameter efficiency (2.82M) and model performance, achieving good classification results on the CIFAR-10 dataset through carefully designed architecture and training strategies.

Experiment

Our experimental process mainly includes experimental setup, experimental methods, experimental results and final discussion.

Experimental Setup

Objective:

- By adding data enhancement, regularization and other strategies, the robustness and accuracy of the original residual extension into the network are improved.

Datasets:

- CIFAR-10 dataset. The dataset contains 60,000 32x32 color images in 10 categories, 6,000 images per category. There are 50,000 training images and 10,000 test images.

Environment:

- Hardware: NVIDIA GeForce RTX 4060 Laptop GPU.
- Total video memory: 8188.0 MB.
- Software: Python programming language and PyTorch framework.

Experimental Methodology

Experimental Design:

- During continuous learning and testing, we found that the following features may affect the prediction accuracy: Mixup&Cutout, ColorJitter, Grayscale, SE Block. Therefore, we set up a control group to compare the effectiveness of each strategy.

Implementation Details:

- Quantitative Analysis: We trained a model that included all the above strategies, and then trained the models after removing Mixup&Cutout, ColorJitter, Grayscale, and SE Block. Finally, we compared the prediction accuracy of each model to show the impact of each strategy on the prediction accuracy.

Variables

Model	Mixup&Cutout	ColorJitter	Grayscale	SE Block
Model 1	Included	Included	Included	Included
Model 2	Unincluded	Included	Included	Included
Model 3	Included	Unincluded	Included	Included
Model 4	Included	Included	Unincluded	Included
Model 5	Included	Included	Included	Unincluded

Table 1: Comparison of different models with feature variations.

Experimental Results

Finally, the accuracy of each model for the unlabeled test set is as follows:

The final results show that the model that includes all strategies performs best, among which the Mixup&Cutout enhancement strategy has the greatest impact on the accuracy. ColorJitter and SE Block have a smaller impact on the

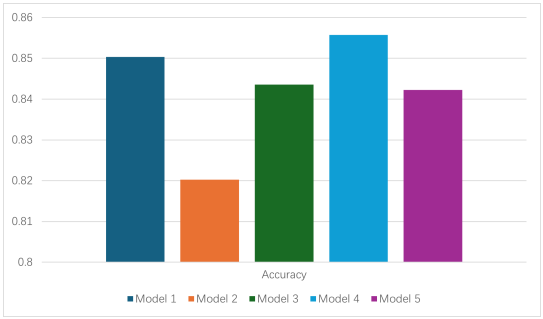

Figure 2: Prediction accuracy of different models.

final prediction accuracy of the model, but they are also indispensable for improving the generalization ability of the model and further optimizing the model. Grayscale has a negative impact on the accuracy of the model.

Error Analysis

Initially, considering the impact of the background color of the image (the main part that must be predicted) on the prediction, we tried to use Grayscale to change all images to gray to try to weaken this impact. However, the results showed that the addition of Grayscale reduced the accuracy of the model. This shows that color is an important feature for distinguishing objects. Therefore, we finally retained the color features of the image, but used Cutout and ColorJitter to weaken the impact of the background color of the image. The final results showed that the prediction accuracy performed well, and the prediction accuracy for the unlabeled dataset reached 0.85547.

Discussion

- The experimental results are in line with expectations. First, the Mixup&Cutout strategy can better increase the difficulty of model training, avoid the model being abnormally sensitive to certain features, and improve the generalization ability of the model.
- ColorJitter can weaken the model's dependence on image color features and avoid misjudgment due to similar color features of images.
- SE Block enables the network to automatically learn the importance weights between channels, enhance key features, and suppress unimportant features, thereby improving the model's expressiveness.
- In the future, we hope to try to use a baseline method to compare and evaluate the performance of new methods, rather than simply setting up a control group. This is an area where the project may need to be improved in the future.

Final Outcome

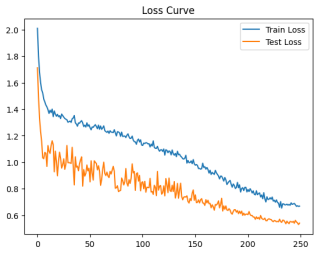

Figure 3: Train and Test Loss

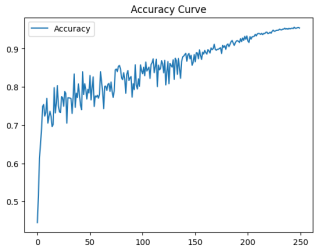

Figure 4: Accuracy change

The training loss of the model drops very stably before the scheduler warm-up is finished. After 10 epochs, the training loss starts to fluctuate. The test loss curve fluctuates significantly at the beginning, but the curve becomes much more stable from the starting point to the end. This indicates that the predictions become increasingly stable in capturing all features.

The accuracy acts as a mirror of the test loss: when the test loss decreases, the accuracy follows a similar pattern. As a result, the two figures have very similar path.

References

- Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.
- DeVries, T., & Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552.
- Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. International Conference on Learning Representations.
- Loshchilov, I., & Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. International Conference on Learning Representations.